UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/779,808	02/18/2004	Paul Anthony Gilkerson	550-525	6826

23117 7590 08/21/2007

NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
FENNEMA, ROBERT E	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
08/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

MAILED

AUG 2 1 2007

Technology Center 2100

Application Number: 10/779,808
Filing Date: February 18, 2004
Appellant(s): GILKERSON, PAUL ANTHONY

Stanley C. Spooner (Reg. No. 27,393)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 5/9/2007 appealing from the Office action mailed 10/16/2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5848269	Hara, Tetsuya	12-1996

Hennessy, John. Patterson, David. "Computer Architecture: A Quantitative Approach". Morgan Kaufmann Publishers Inc, Third Edition. May 17, 2002. Pages A-2 and A-3.

Furber, Steve. "ARM System-on-Chip Architecture". Addison-Wesley, 2000. Pages 382-384, 387-388.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-6, 10-16, and 20-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Furber, in view of Patterson.

3. As per Claim 1, Furber teaches: A data processing apparatus, comprising:

a processor operable to execute a stream of instructions (Page 387, the AMULET 3);

a prefetch unit operable to prefetch instructions from a memory prior to sending those instructions to the processor for execution (Page 387), the prefetch unit being operable to receive from the memory simultaneously a plurality of prefetched instructions from sequential addresses in memory (Page 387, it can fetch two Thumb instructions), and being operable to detect whether any of those prefetched instructions are an instruction flow changing instruction, and based thereon to output a fetch address for a next instruction to be prefetched by the prefetch unit (Page 382, where it says it modifies the predicted control flow to the predicted target address. Page 387 says that the prefetch unit in the AMULET3 functions similar to the jump trace buffer shown on 382, but to support the multiple instructions, and that a "hit" indicates that it is a branch instruction); and

address generation logic within the prefetch unit and responsive to a selected prefetched instruction that is detected to be said instruction flow changing instruction, for determining a target address to be output as the fetch address (Pages 382-383), the address generation logic having a first address generation path for determining the

target address if the selected prefetched instruction is a first prefetched instruction in said plurality (Page 388), and at least one further address generation path for determining the target address if the selected prefetched instruction is one of the other prefetched instructions in said plurality, the first prefetched instruction being earlier in said stream than said other prefetched instructions (Page 388), the first address generation path generating the target address more quickly than the at least one other further address generation path (Page 388, the first path takes priority); and

whereby, in the event that the first prefetched instruction is said selected prefetched instruction, the prefetch unit outputs the associated target address as the fetch address earlier than if one of said other prefetched instructions is said selected prefetched instruction (Page 388, the first path takes priority), but fails to teach:

a pipeline stage, provided in said at least one further address generation path, for increasing generation speed of the target address by the first address generation path.

However, Patterson teaches that pipelining a processor, or adding stages of a pipeline, will reduce CPI, and reduce clock cycle time (increasing the rate at which the clock runs) (Page A-3), which would then speed up execution of the first address path, due to the increased clock rate. Given this advantage, it would have been obvious to one of ordinary skill in the art at the time the invention was made to pipeline a processing path that took one long clock cycle, to increase the clock rate for other paths, and apply it to Furber's invention.

4. As per Claim 2, Furber teaches: A data processing apparatus as claimed in claim 1, further comprising:

prediction logic operable to predict, for a prefetched instruction whose execution is conditional, whether that conditional prefetched instruction will be executed by the processor (Pages 382-383);

in the event that the first prefetched instruction is said selected prefetched instruction and its execution is conditional, the prefetch unit being operable to output the associated target address as the fetch address if the prediction logic predicts that the first prefetched instruction will be executed by the processor (Pages 382-383).

5. As per Claim 3, Furber teaches: A data processing apparatus as claimed in claim 1, wherein the prefetch unit associates a different priority level with each of the plurality of prefetched instructions, the first prefetched instruction having the highest priority level, and if more than one of the plurality of prefetched instructions is detected to be said instruction flow changing instruction, the prefetch unit is operable to determine as said selected prefetched instruction the prefetched instruction having the higher priority level from those prefetched instructions detected to be said instruction flow changing instruction, whereby the target address associated with that selected prefetched instruction is output as the fetch address (Page 388).

6. As per Claim 4, Furber teaches: A data processing apparatus as claimed in claim 1, wherein the address generation logic is operable to generate the target address for

the first prefetched instruction in a same clock cycle as the prefetch unit detects that that first prefetched instruction is said instruction flow changing instruction (Page 382).

7. As per Claim 5, Furber teaches: A data processing apparatus as claimed in claim 1, wherein predetermined logic is shared between the first address generation path and the at least one further address generation path (Page 387, a memory).

8. As per Claim 6, Furber teaches: A data processing apparatus as claimed in claim 1, wherein the at least one further address generation path comprises a single further address generation path used to determine the target address for any prefetched instructions other than said first prefetched instruction (Pages 387-388).

9. As per Claim 10, Furber teaches: A data processing apparatus as claimed in claim 1, wherein if none of the plurality of prefetched instructions is said instruction flow changing instruction, the prefetch unit is operable to generate the fetch address by incrementing a previous fetch address output by the prefetch unit (Page 382, figure 14.6).

10. As per Claim 11, Furber teaches: A method of operating a data processing apparatus to determine a target address for an instruction flow changing instruction, the data processing apparatus having a processor operable to execute a stream of instructions (Page 387, the AMULET 3), and a prefetch unit operable to prefetch

instructions from a memory prior to sending those instructions to the processor for execution (Page 387), and to output a fetch address for a next instruction to be prefetched from the memory (Page 382, where it says it modifies the predicted control flow to the predicted target address. Page 387 says that the prefetch unit in the AMULET3 functions similar to the jump trace buffer shown on 382, but to support the multiple instructions, and that a "hit" indicates that it is a branch instruction), the method comprising the steps of:

(a) receiving from the memory simultaneously a plurality of prefetched instructions from sequential addresses in memory (Page 387, it can fetch two Thumb instructions);

(b) detecting whether any of those prefetched instructions are an instruction flow changing instruction (Page 382, where it says it modifies the predicted control flow to the predicted target address. Page 387 says that the prefetch unit in the AMULET3 functions similar to the jump trace buffer shown on 382, but to support the multiple instructions); and

(c) for a selected prefetched instruction that is detected to be said instruction flow changing instruction, determining a target address to be output as the fetch address (Pages 382-383) by performing one of the steps of:

(c)(1) employing a first address generation path to determine the target address if the selected prefetched instruction is a first prefetched instruction in said plurality (Page 388); or

(c)(2) employing at least one further address generation path to determine the target address if the selected prefetched instruction is one of the other prefetched instructions in said plurality (Page 388); the first prefetched instruction being earlier in said stream than said other prefetched instructions, and the first address generation path being arranged to generate the target address more quickly than the at least one other further address generation path (Page 388, the first path takes priority);

(e) outputting as the fetch address the target address generated at step (c); whereby, in the event that the first prefetched instruction is said selected prefetched instruction, the prefetch unit is operable to output the associated target address as the fetch address earlier than if one of said other prefetched instructions is said selected prefetched instruction (Page 388, the first path takes priority), but fails to teach:

(d) providing a pipeline stage in the at least one further address generation path in order to increase speed of generation of the target address by the first address generation path.

However, Patterson teaches that pipelining a processor, or adding stages of a pipeline, will reduce CPI, and reduce clock cycle time (increasing the rate at which the clock runs) (Page A-3), which would then speed up execution of the first address path, due to the increased clock rate. Given this advantage, it would have been obvious to one of ordinary skill in the art at the time the invention was made to pipeline a processing path that took one long clock cycle, to increase the clock rate for other paths, and apply it to Furber's invention.

11. As per Claim 12, Furber teaches: A method as claimed in claim 11, further comprising the step of:

employing prediction logic to predict, for a prefetched instruction whose execution is conditional, whether that conditional prefetched instruction will be executed by the processor (Pages 382-383);

in the event that the first prefetched instruction is said selected prefetched instruction and its execution is conditional, the prefetch unit being operable to output the associated target address as the fetch address if the prediction logic predicts that the first prefetched instruction will be executed by the processor (Pages 382-383).

12. As per Claim 13, Furber teaches: A method as claimed in claim 11, further comprising the steps of:

associating a different priority level with each of the plurality of prefetched instructions, the first prefetched instruction having the highest priority level (Page 388);

if more than one of the plurality of prefetched instructions is detected to be said instruction flow changing instruction, determining as said selected prefetched instruction for said step (c) the prefetched instruction having the higher priority level from those prefetched instructions detected to be said instruction flow changing instruction (Page 388);

whereby at said step (d) the target address associated with that selected prefetched instruction is output as the fetch address (Page 388).

13. As per Claim 14, Furber teaches: A method as claimed in claim 11, wherein at said step (c)(1) the target address for the first prefetched instruction is generated in a same clock cycle that, during said step (b), that first prefetched instruction is detected as said instruction flow changing instruction (Page 382).

14. As per Claim 15, Furber teaches: A method as claimed in claim 11, wherein predetermined logic is shared between the first address generation path and the at least one further address generation path (Page 387, a memory).

15. As per Claim 16, Furber teaches: A method as claimed in claim 11, wherein the at least one further address generation path comprises a single further address generation path used at said step (c)(2) to determine the target address for any prefetched instructions other than said first prefetched instruction (Pages 387-388).

16. As per Claim 20, Furber teaches: A method as claimed in claim 11, wherein if none of the plurality of prefetched instructions is determined at said step (b) to be said instruction flow changing instruction, the method further comprises the step of:

generating the fetch address by incrementing a previous fetch address output by the prefetch unit, and outputting that fetch address at said step (d) (Page 382, figure 14.6).

17. As per Claim 21, Furber teaches: A prefetch unit for a data processing apparatus that has a processor operable to execute a stream of instructions (Page 387, the AMULET 3), the prefetch unit being operable to prefetch instructions from a memory prior to sending those instructions to the processor for execution (Page 387), the prefetch unit being operable to receive from the memory simultaneously a plurality of prefetched instructions from sequential addresses in memory (Page 387, it can fetch two Thumb instructions), and being operable to detect whether any of those prefetched instructions are an instruction flow changing instruction, and based thereon to output a fetch address for a next instruction to be prefetched by the prefetch unit (Page 382, where it says it modifies the predicted control flow to the predicted target address. Page 387 says that the prefetch unit in the AMULET3 functions similar to the jump trace buffer shown on 382, but to support the multiple instructions, and that a "hit" indicates that it is a branch instruction), the prefetch unit comprising:

address generation logic, responsive to a selected prefetched instruction that is detected to be said instruction flow changing instruction, for determining a target address to be output as the fetch address (Pages 382-383), the address generation logic having a first address generation path for determining the target address if the selected prefetched instruction is a first prefetched instruction in said plurality (Page 388), and at least one further address generation path for determining the target address if the selected prefetched instruction is one of the other prefetched instructions in said plurality (Page 388), the first prefetched instruction being earlier in said stream than said other prefetched instructions, the first address generation path generating the

target address more quickly than the at least one other further address generation path (Page 388, the first path takes priority); and

whereby, in the event that the first prefetched instruction is said selected prefetched instruction, the prefetch unit is operable to output the associated target address as the fetch address earlier than if one of said other prefetched instructions is said selected prefetched instruction (Page 388, the first path takes priority), but fails to teach:

a pipeline stage, provided in said at least one further address generation path, for increasing generation speed of the target address by the first address generation path.

However, Patterson teaches that pipelining a processor, or adding stages of a pipeline, will reduce CPI, and reduce clock cycle time (increasing the rate at which the clock runs) (Page A-3), which would then speed up execution of the first address path, due to the increased clock rate. Given this advantage, it would have been obvious to one of ordinary skill in the art at the time the invention was made to pipeline a processing path that took one long clock cycle, to increase the clock rate for other paths, and apply it to Furber's invention.

18. Claims 7-9 and 17-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Furber and Patterson, further in view of Hara (USPN 5,848,269).

19. As per Claim 7, Furber teaches: A data processing apparatus as claimed in claim 1, wherein the prefetch unit comprises decode logic operable to detect whether any of

the plurality of prefetched instructions are said instruction flow changing instruction (Pages 382-383), but fails to explicitly teach:

the decode logic further being operable to decode from each prefetched instruction detected to be said instruction flow changing instruction an immediate value to be input to the address generation logic.

While Furber teaches a prefetch unit to detect whether there is an instruction flow changing instruction using a branch prediction unit, and outputting a target address, it is not explicitly taught that the branch prediction unit can handle branches with immediate values. However, Hara teaches a branch prediction mechanism, which can be used for effectively calculating branch targets, without adding excessive hardware (Column 5, Lines 18-24), capable of effectively predicting without a correlating past history (Column 5, Lines 1-5), which has a method for predicting branches with an immediate value, and outputting a target address (Column 8, Lines 44-63, Column 19, Line 58 – Column 20, Line 4, also see Figure 15). Given that branches with immediate targets exist and can be encountered in the instruction stream, and given a need to handle them in Furber's invention, one of ordinary skill in the art at the time the invention was made would have recognized the advantage of using a branch prediction unit such as Hara's, to take advantage of handling immediate values in a branch, while minimizing extra hardware, and being able to effectively predict branches even without correlation with past results.

20. As per Claim 8, Hara teaches: A data processing apparatus as claimed in claim 7, wherein the address generation logic comprises adder logic operable to determine

the target address for the selected prefetched instruction by adding the associated input immediate value to the address of that selected prefetched instruction (Figure 15, and Column 19, Line 58 – Column 20, Line 4).

21. As per Claim 9, Furber and Hara teach: A data processing apparatus as claimed in claim 8, wherein the adder logic is shared between the first address generation path and the at least one further address generation path (Hara, Column 19, Line 58 – Column 20, Line 4 discloses the adder, and Furber, Page 388 discloses that one path takes priority, meaning only one target can be output at a time, meaning the adder must be shared).

22. As per Claim 17, Furber teaches: A method as claimed in claim 11, wherein the prefetch unit comprises decode logic operable at said step (b) to detect whether any of the plurality of prefetched instructions are said instruction flow changing instruction (Pages 382-383), but fails to explicitly teach: the method further comprising the step of:

employing the decode logic to decode from each prefetched instruction detected to be said instruction flow changing instruction an immediate value for use in said step (c).

While Furber teaches a prefetch unit to detect whether there is an instruction flow changing instruction using a branch prediction unit, and outputting a target address, it is not explicitly taught that the branch prediction unit can handle branches with immediate values. However, Hara teaches a branch prediction mechanism, which can be used for

effectively calculating branch targets, without adding excessive hardware (Column 5, Lines 18-24), capable of effectively predicting without a correlating past history (Column 5, Lines 1-5), which has a method for predicting branches with an immediate value, and outputting a target address (Column 8, Lines 44-63, Column 19, Line 58 – Column 20, Line 4, also see Figure 15). Given that branches with immediate targets exist and can be encountered in the instruction stream, and given a need to handle them in Furber's invention, one of ordinary skill in the art at the time the invention was made would have recognized the advantage of using a branch prediction unit such as Hara's, to take advantage of handling immediate values in a branch, while minimizing extra hardware, and being able to effectively predict branches even without correlation with past results.

23. As per Claim 18, Hara teaches: A method as claimed in claim 17, wherein at said step (c) the target address for the selected prefetched instruction is determined by adding the associated immediate value to the address of that selected prefetched instruction (Figure 15, and Column 19, Line 58 – Column 20, Line 4).

24. As per Claim 19, Furber and Hara teach: A method as claimed in claim 18, wherein adder logic used to perform said adding step is shared between the first address generation path and the at least one further address generation path (Hara, Column 19, Line 58 – Column 20, Line 4 discloses the adder, and Furber, Page 388 discloses that one path takes priority, meaning only one target can be output at a time, meaning the adder must be shared).

(10) Response to Argument

25. Appellant has argued in Section A that Furber does not teach a determination between two address generation paths based upon the IFCI being a first prefetched instruction. Appellant is correct when stating that the Examiner interpreted the first, even half to be one path, and the second, odd half to be the second path. Appellant has further argued that any even addressed instruction is handled by the first path, and any odd addressed instruction is handled by the second path, which Examiner also agrees with. However, Examiner disagrees with the Appellant in the remaining arguments, that Furber does not identify an IFCI being first or not, because the even half is the "first". Zero comes before One, Two comes before Three. There is a further indication in Furber that shows why the even value is considered to be "first" which Examiner will address momentarily. Appellant has request where Furber teaches a determination that an instruction is an IFCI, and Examiner refers to Page 388, where the IFCI is a branch instruction. Branch instructions change the flow of the instruction stream to the target of the branch, thus is an instruction flow changing instruction. Since Page 388 shows that it needs to be determined if there are 0, 1, or 2 branches in the "halves", Examiner believes it is quite clear that Furber can detect these branches, and determine which one is "first" (ie, if one is in the even half or not). As for there being two different address generation paths, Examiner will address this in further detail in the next paragraph addressing Argument B.

26. In Section B, Appellant has argued that Furber does not teach the limitation of the first path generated the target address "more quickly" than the other path. Despite Appellants statement to the contrary, Examiner very much understands the claim language and the reference, and believes that it is quite obvious that priority is the same as generating "more quickly", if one actually looks at the reference, and how the Examiner laid out the rejection in the previous actions. In fact, Examiner believes that the Appellant misunderstands the Furber reference. As Examiner has pointed out before, despite the Furber system being able to fetch up to two instructions at a time, in the case that they are both branches, both branches are not predicted at the same time. Furber states on Page 388 that the even (first) branch takes priority, and this very clear language should indicate to one of skill in the art that the branch prediction unit can only address one of the branches at a time, if both instructions are in fact branches. Therefore, it should be quite clear that if the even instruction is taking priority in the branch prediction unit, that it will generate its address before the odd instruction that has not yet been able to use the branch predictor. Thus, priority is very clearly the same as generating "more quickly", as the priority allows it to execute before the other path, and if one instruction path is executing, and the other is not, then there is absolutely no way to reasonably suggest that the two paths will generate their addresses at the same time, because that is clearly impossible in light of the reference. If Appellant maintains that the two paths will both use the branch prediction unit, and generate their addresses at the same time, then the Examiner would ask the Appellant what priority is supposed to mean in this case. Since priority can only mean one thing, Examiner submits that his

interpretation of the art is correct, and therefore, quite clearly reads on the claim language. Appellant appears to be arguing that both of his paths are executing simultaneously, and that the first path can generate the target address faster than the other, when they are both executing at the same time, however, this is not even remotely suggested in the claim language, thus the "more quickly" limitation is quite broad, and thus Furber reads on it. As for Furber not reading on the limitation of "a pipeline stage, provided in said one further address generation path, for increasing generation speed of the target address by the first address generation path", this is hardly an admission that Furber does not teach choosing among data paths (which Examiner would like to point out, is not claimed, or even remotely implied), but rather showing that Furber does not explicitly teach pipelines, thus Patterson was brought in to introduce pipelining, one of the most basic and fundamental concepts in processor design over the last few decades. By pipelining a machine, the clock rate of the system as a whole can be drastically increased, thus resulting in all pipeline stages receiving a significant speed boost. It is for this reason that Patterson was used, and any attempt to read further into this combination would be erroneous, as Examiner has already laid out above how Furber clearly teaches the concept of one path generating an address "more quickly" than another path, thus there is no need for Patterson to teach such a limitation.

27. Regarding Appellant's Argument C, Examiner believes that the explanation above for Section B addresses all the arguments presented, as Appellant has misinterpreted the usage of Patterson in the rejection. Again, Appellant here attempts to

argue limitations that are not present in the claims, namely, deciding between two paths. The only language in the claims that could even remotely suggest this limitation is the "employing" of a path, but to say that "employing" a path is the same as "deciding" is not the broadest reasonable interpretation of the claim language, and Examiner would submit that it is not a reasonable interpretation of the language even if there is an attempt to read the language more narrowly than it should be, thus, even if the Examiner was to improperly examine the claims, Examiner believes it is unreasonable to take the interpretation that the Appellant has attempted to argue, and therefore the fact that Furber or Patterson allegedly do not teach these features is immaterial.

28. Regarding Appellants Argument D, again, Examiner has already covered this argument in Section B, and refers to his remarks on Section B for these arguments as well.

29. Regarding Argument E, Examiner again refers to his statements for Section B for the bulk of this argument. Appellant has argued that Patterson's pipelining is somehow a "imagined improvement", however Examiner strongly disagrees with this statement. As stated before, pipelining is one of the most basic and fundamental concepts of modern computing, to the point where it is borderline inherent. Patterson is a textbook which discloses pipelining, and anyone of any skill in the art would recognize the significant advantages pipelining has over non-pipelined machines. To even suggest that pipelining is an "imagined improvement" is a gross misinterpretation of the

references, and the art in general. Examiner provided more than enough evidence in the rejection to show why pipelining is extremely beneficial, and why one of ordinary skill in the art would have been motivated to use it. Additionally, the Examiner notes the Supreme Court Decision in *KSR International Co. v. Teleflex Inc.,* which has determined that it is obvious in the art to apply a known technique to a known device ready for improvement to yield predicable results. Given that pipelining is so fundamental to modern computers, one of ordinary skill in the art would have been motivated to apply the extremely well known teachings of pipelining to Furber's invention, to achieve the predicable results of significant faster clock speeds and instruction throughput.

30. Regarding Argument F, again, Examiner refers back to his remarks towards Argument B to explain how Furber teaches the claimed invention, thus, it is quite impossible for Furber to teach away from the invention when Furber himself teaches the invention.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Robert Fennema

Conferees:

Matt Kim

MATTHEW KIM
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Lynne Browne

Appeal Practice Specialist, TQAS

Technology Center 2100